A Fast-but-Gentle Introduction to Artificial Intelligence Acceleration

Dr. **Francesco Conti**, DEI & ARCES
f.conti@unibo.it

Deep NN Timeline

[Yann LeCun, ISSCC 2019]

- **1940s: Neural networks were proposed**
- **1960s: Deep neural networks were proposed**
- **1989: Neural network for recognizing digits (LeNet)**
- **1990s: Hardware for shallow neural nets**
  - Example: Intel ETANN (1992)

NVIDIA GPUs with CUDA available

- **2011: Breakthrough DNN-based speech recognition**
  - Microsoft real-time speech translation
- **2012: DNNs for vision supplanting traditional ML**
  - AlexNet for image classification
- **2014+: Rise of DNN accelerator research**
  - Examples: Neuflow, DianNao, etc.

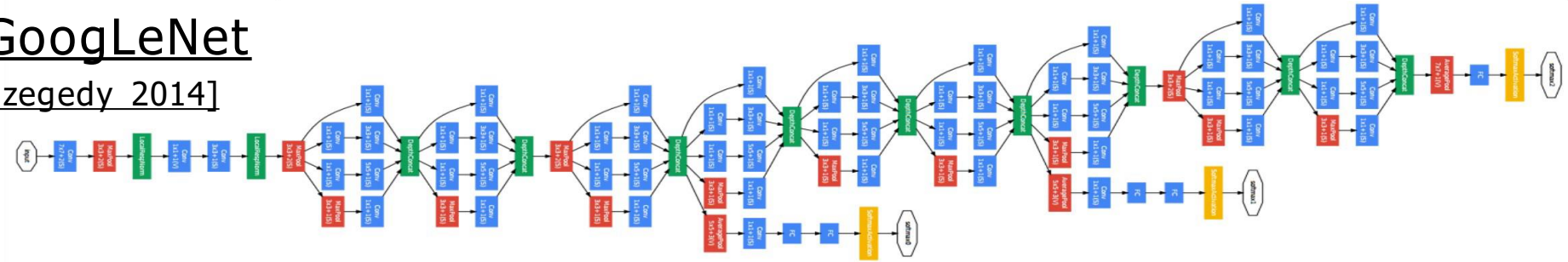
Deep ConvNets (depth inflation)

VGG

[Simonyan 2013]

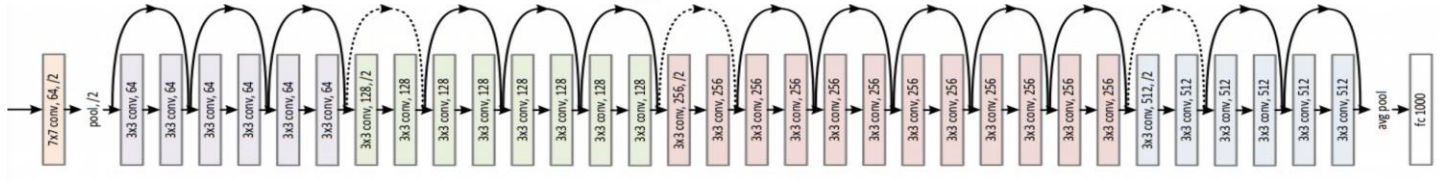
GoogLeNet

Szegedy 2014]



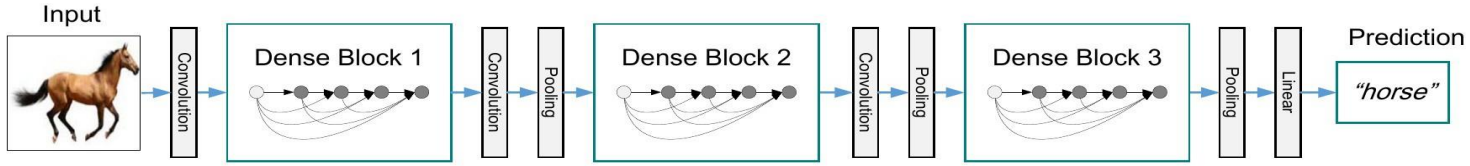
ResNet

[He et al. 2015]

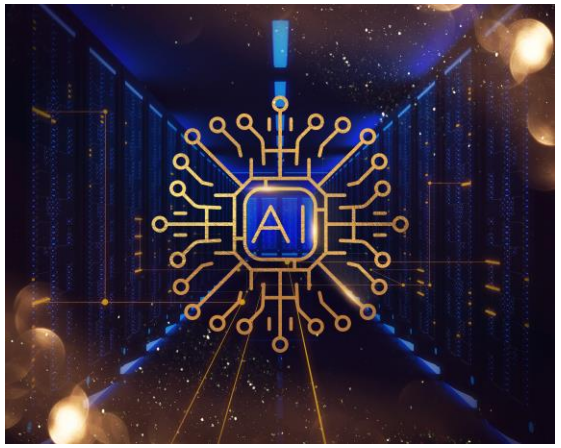


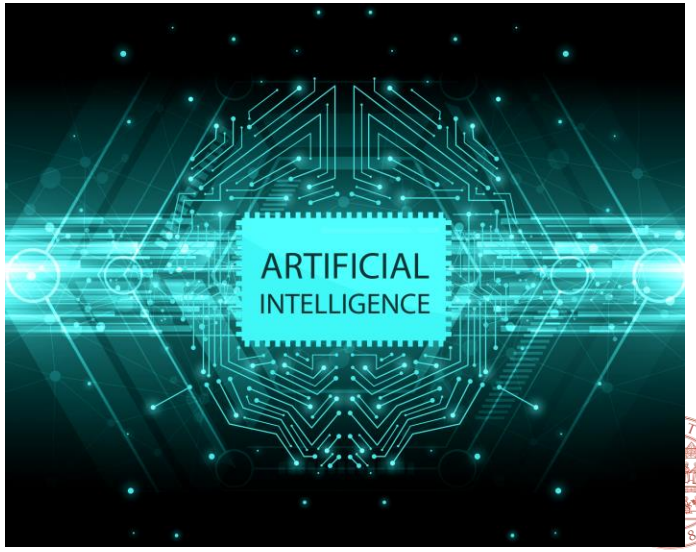
DenseNet

[Huang et al 2017]



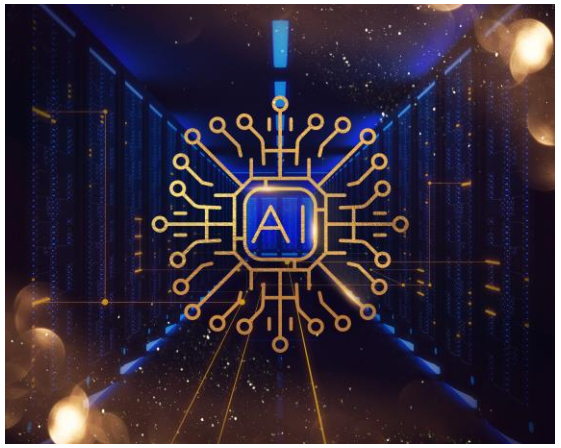
Searching for "AI" on Google Image Search



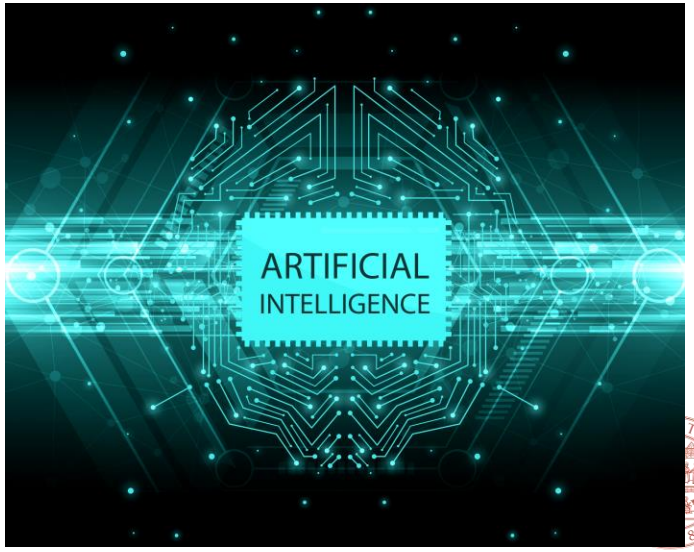


[CC images, various sources]

Searching for "AI" on Google Image Search

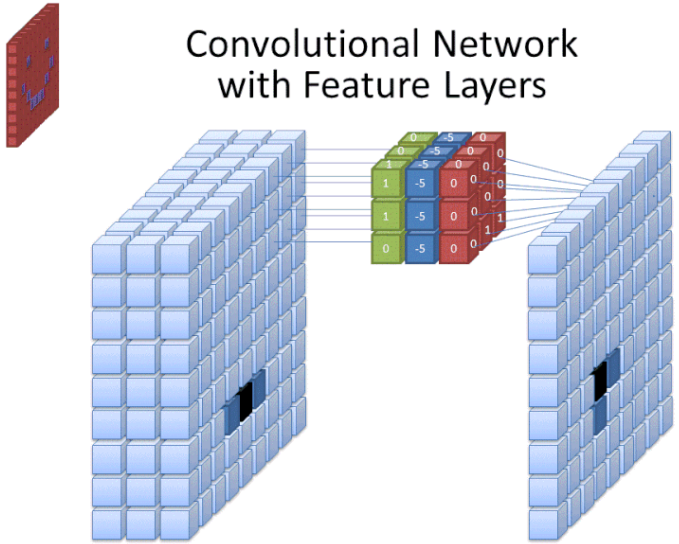


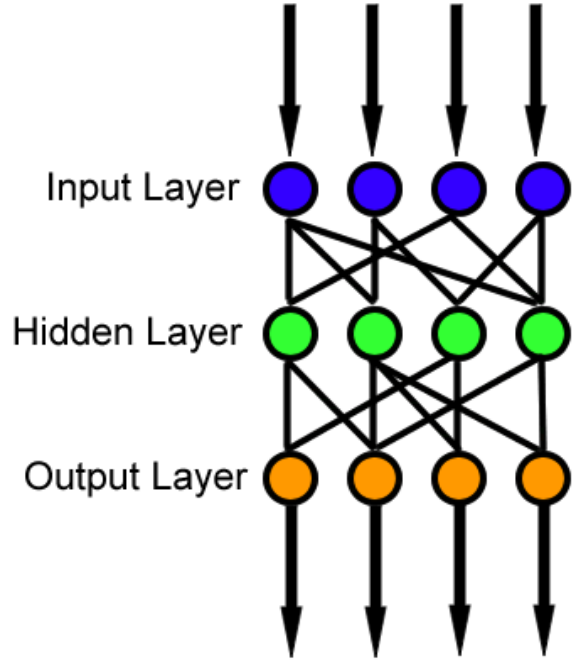
Ok, not much information here!

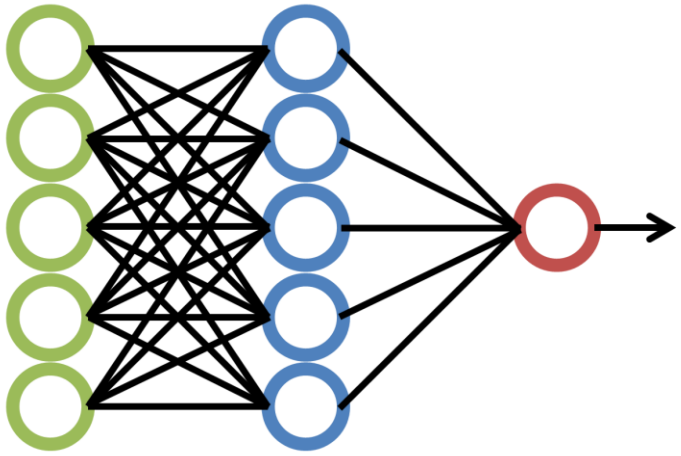


[CC images, various sources]

Searching for "Deep Neural Network" on Google Image Search

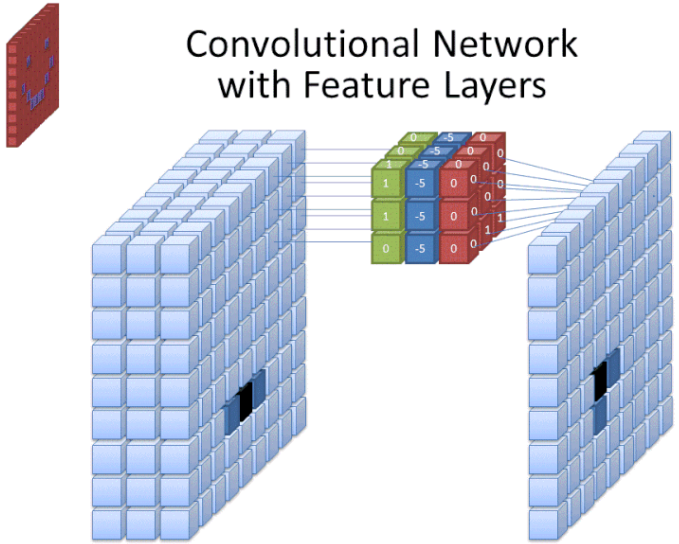




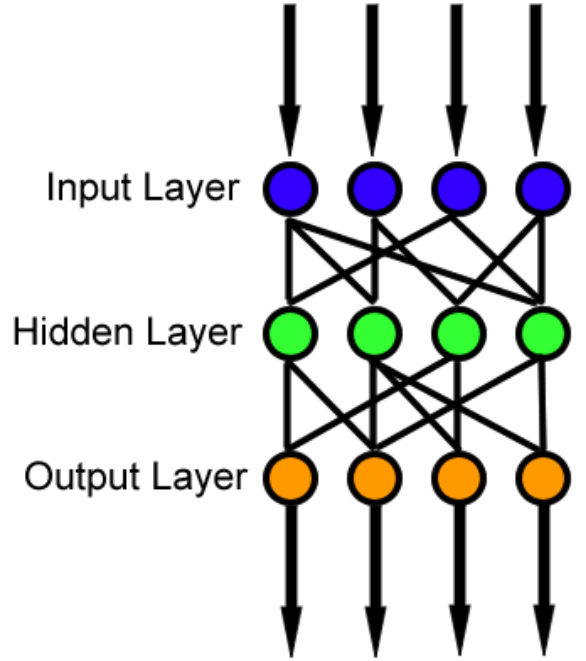


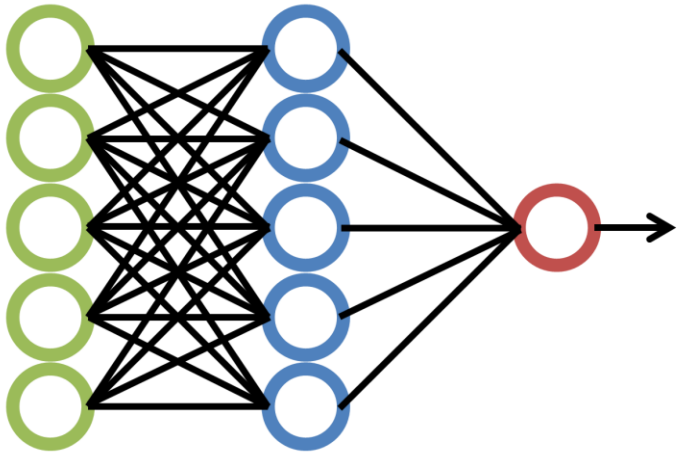
[CC images, various sources]

Searching for "Deep Neural Network" on Google Image Search



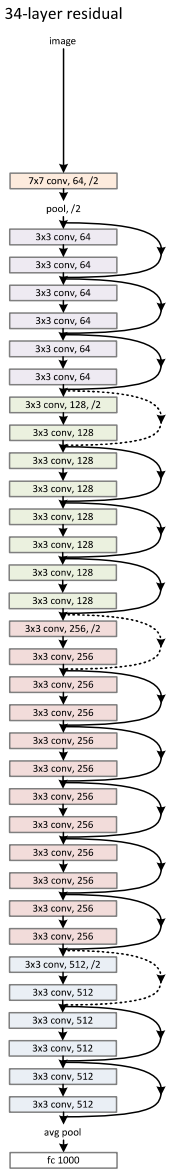
Much better!
But still not crystal clear.



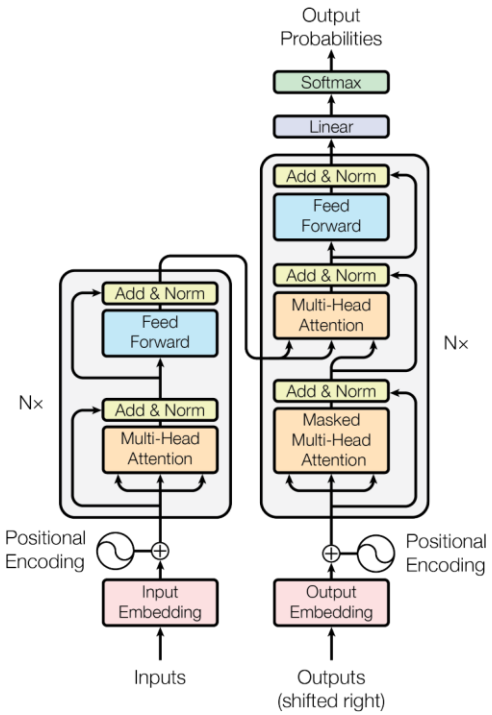


[CC images, various sources]

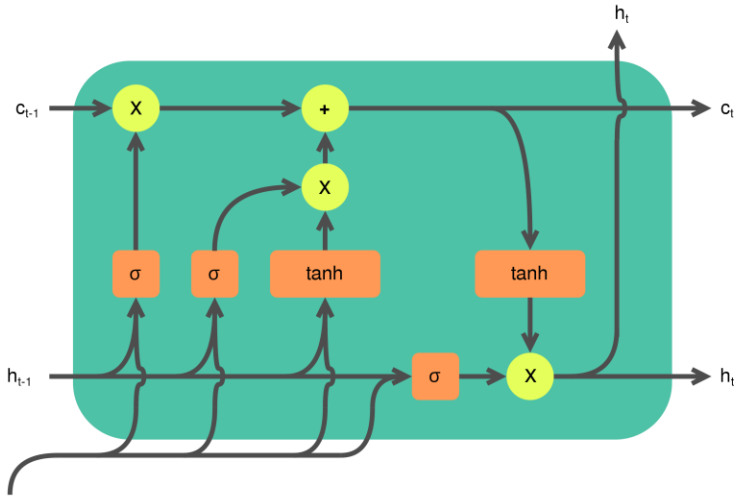
Looking inside papers!



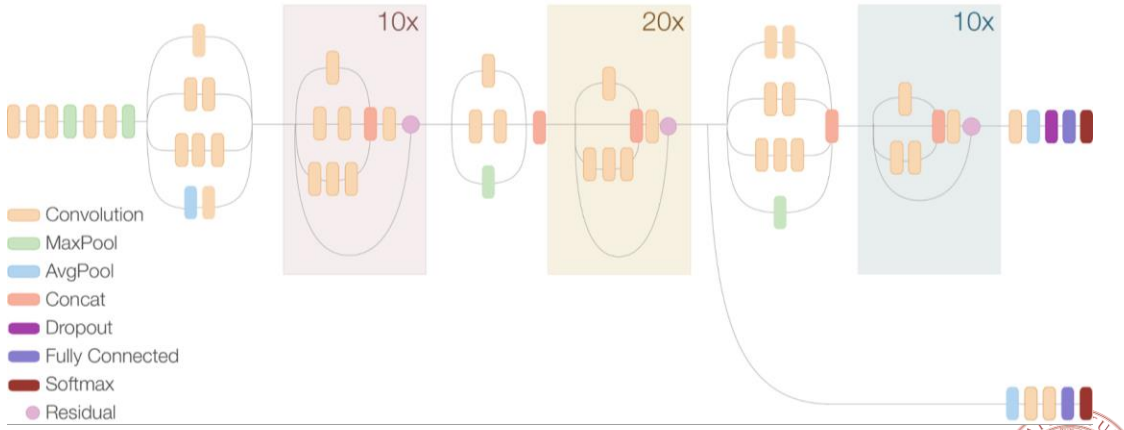
[ResNet-34, He et al., arXiv:1512.03385]



[Transformer, Vaswani et al., arXiv:1706.03762]



[LSTM layer, image from Wikipedia, CC BY 4.0 Guillaume Chevalier]

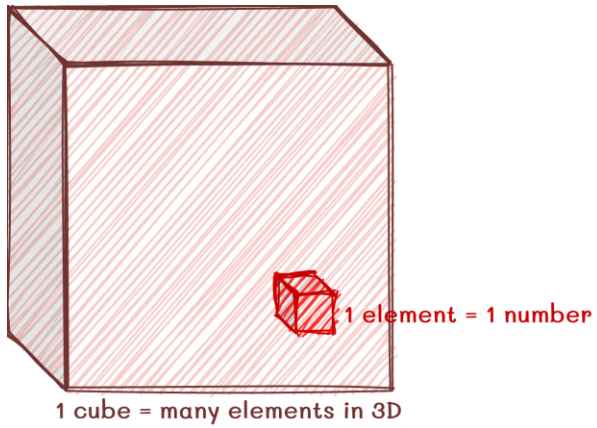


[Inception-ResNet v2, https://ai.googleblog.com/2016/08/improving-inception-and-image.html]

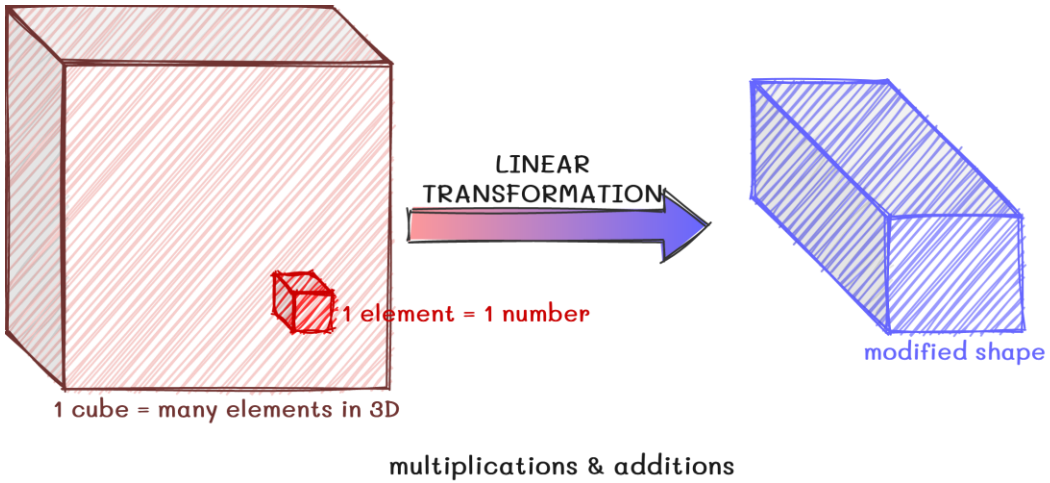
Deep Learning from the Eye of an Accelerator Architect

1 element = 1 number

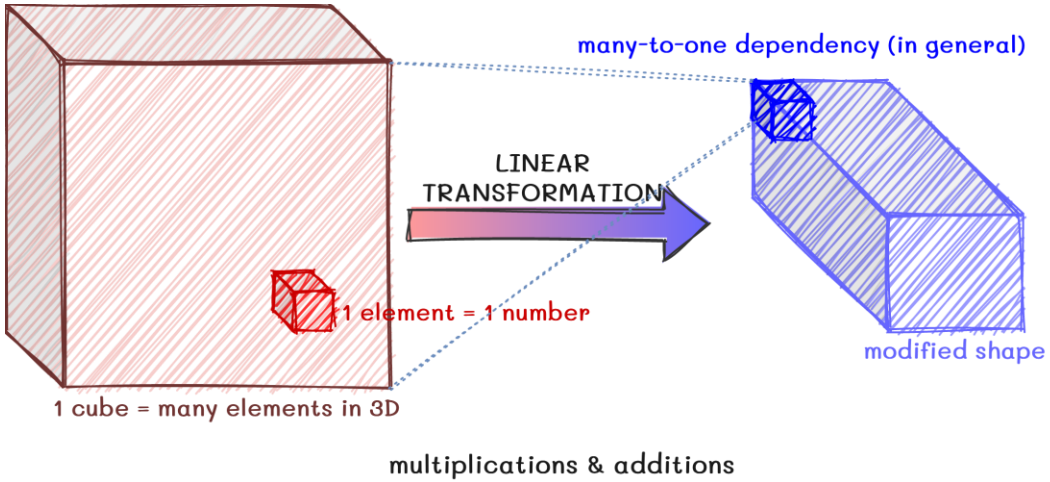
Deep Learning from the Eye of an Accelerator Architect



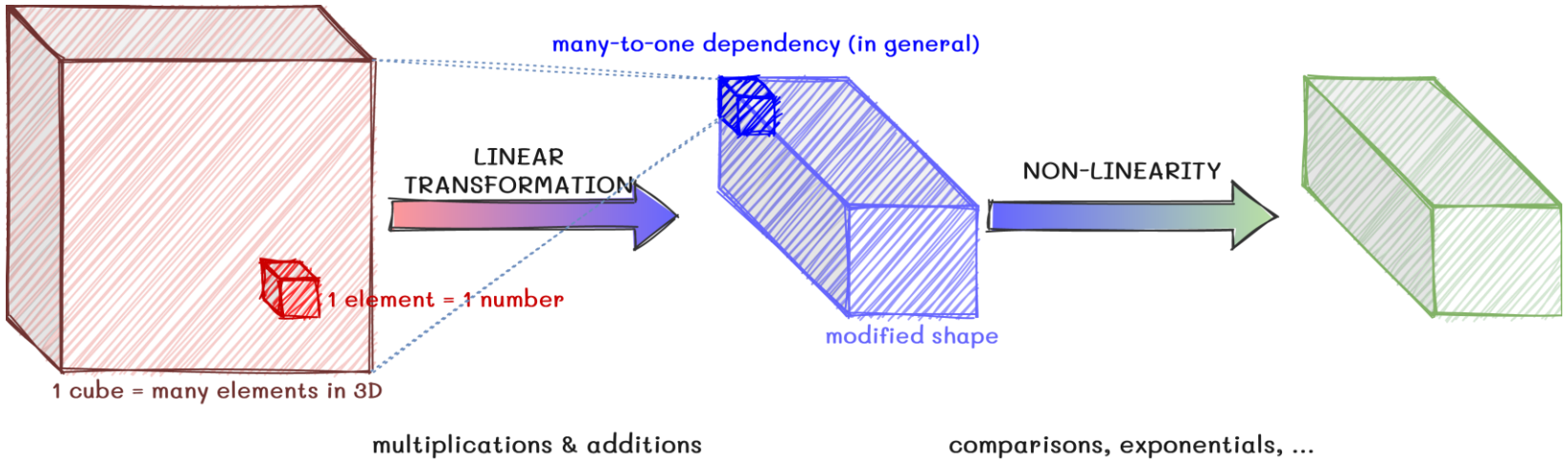
Deep Learning from the Eye of an Accelerator Architect



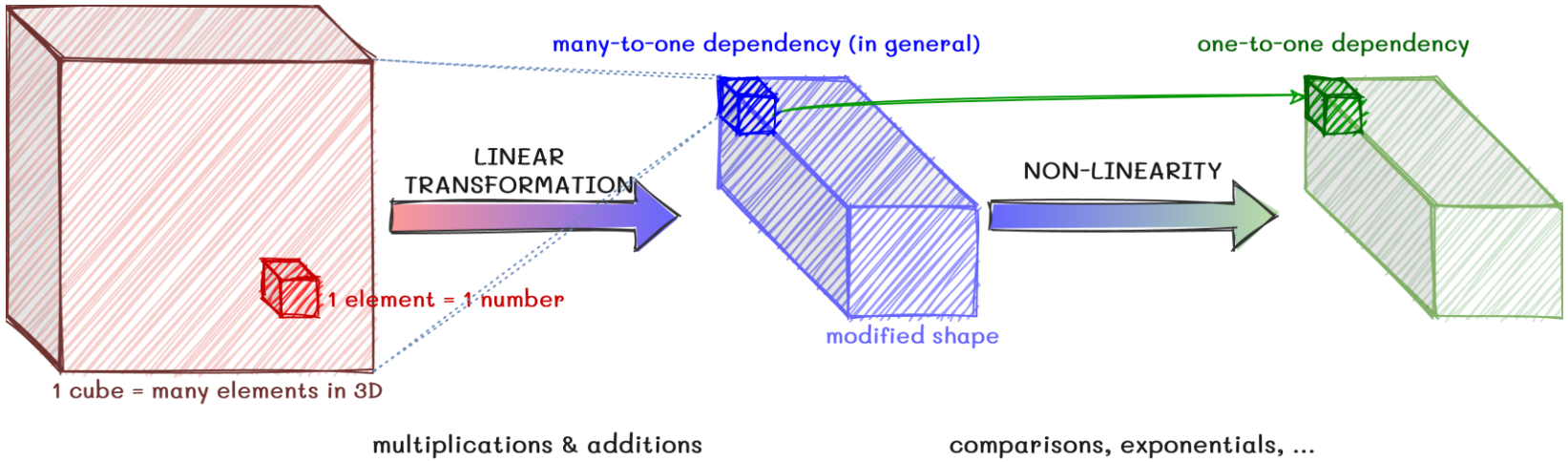
Deep Learning from the Eye of an Accelerator Architect



Deep Learning from the Eye of an Accelerator Architect

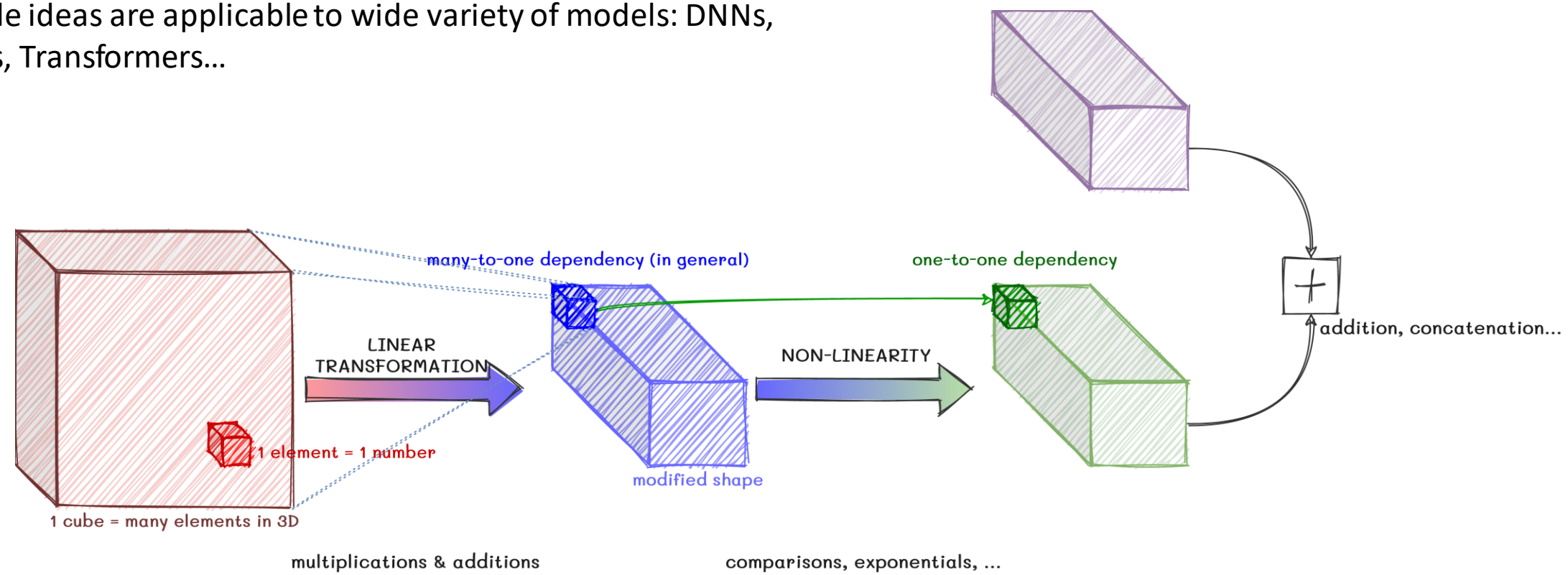


Deep Learning from the Eye of an Accelerator Architect



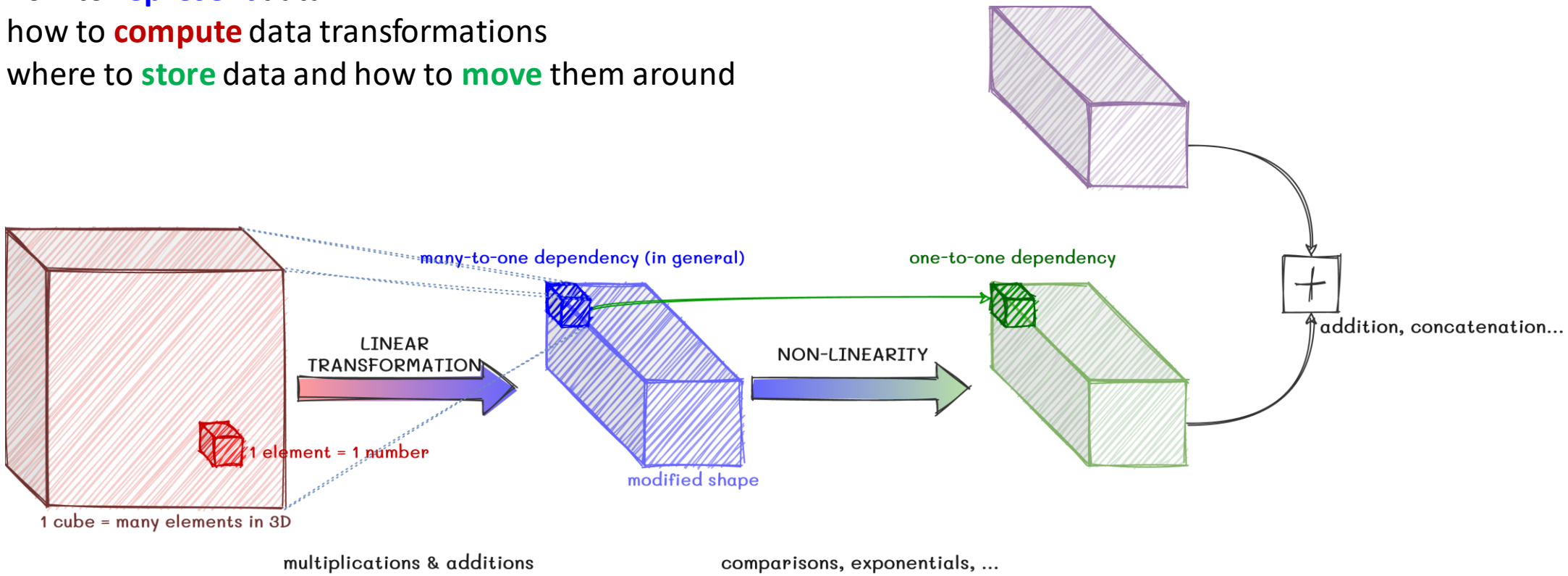
Deep Learning from the Eye of an Accelerator Architect

Infinite compositions, but “basic ingredients” are all here!
Simple ideas are applicable to wide variety of models: DNNs, RNNs, Transformers…



Deep Learning from the Eye of an Accelerator Architect

1. how to **represent** data
2. how to **compute** data transformations
3. where to **store** data and how to **move** them around



Deep Learning from the Eye of a Accelerator Architect

1. how to **represent** data

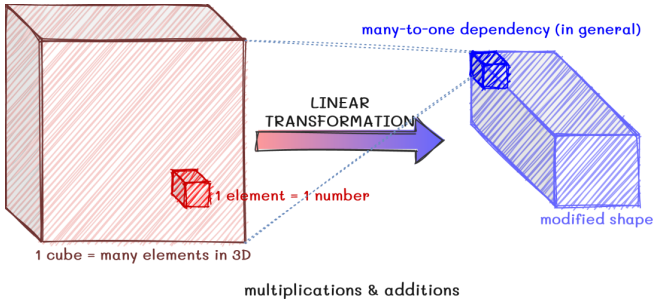
Method	Data Range	Parameter Range	ImageNet Top1 Acc
Full-Precision	Real numbers (FP32)	Real numbers (FP32)	69.6
Linear Quantization	Integers 0 to 255	Integers -128 to +127	69.6
PACT	Integers 0 to 15	Integers -8 to +7	69.2
PACT	Integers 0 to 7	Integers -4 to +3	68.1
PACT	Integers 0 to 3	Integers -2 to +1	64.4
PACT	Integers 0 to 3	-1 or +1	62.9
XNOR-Net	-1 or +1	-1 or +1	51.2

Deep Learning from the Eye of a Accelerator Architect

1. how to **represent** data

Method	Data Range	Parameter Range	ImageNet Top1 Acc
Full-Precision	Real numbers (FP32)	Real numbers (FP32)	69.6
Linear Quantization	Integers 0 to 255	Integers -128 to +127	69.6
PACT	Integers 0 to 15	Integers -8 to +7	69.2
PACT	Integers 0 to 7	Integers -4 to +3	68.1
PACT	Integers 0 to 3	Integers -2 to +1	64.4
PACT	Integers 0 to 3	-1 or +1	62.9
XNOR-Net	-1 or +1	-1 or +1	51.2

2. how to **compute** data transformations



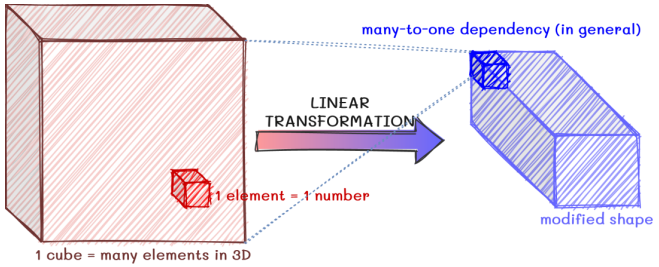
i. dominated by simple arithmetic operations
ii. many-to-one → many possible compute orderings
iii. independent operations → can be done in parallel / hierarchically
iv. hardware complexity / speed /energy related to representation

Deep Learning from the Eye of a Accelerator Architect

1. how to **represent** data

Method	Data Range	Parameter Range	ImageNet Top1 Acc
Full-Precision	Real numbers (FP32)	Real numbers (FP32)	69.6
Linear Quantization	Integers 0 to 255	Integers -128 to +127	69.6
PACT	Integers 0 to 15	Integers -8 to +7	69.2
PACT	Integers 0 to 7	Integers -4 to +3	68.1
PACT	Integers 0 to 3	Integers -2 to +1	64.4
PACT	Integers 0 to 3	-1 or +1	62.9
XNOR-Net	-1 or +1	-1 or +1	51.2

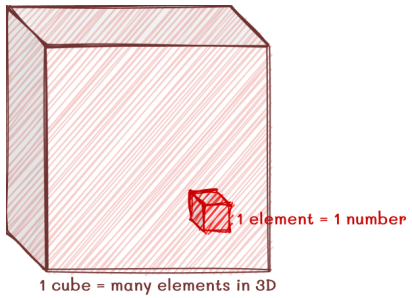
2. how to **compute** data transformations



multiplications & additions

i. dominated by simple arithmetic operations
ii. many-to-one → many possible compute orderings
iii. independent operations → can be done in parallel / hierarchically
iv. hardware complexity / speed /energy related to representation

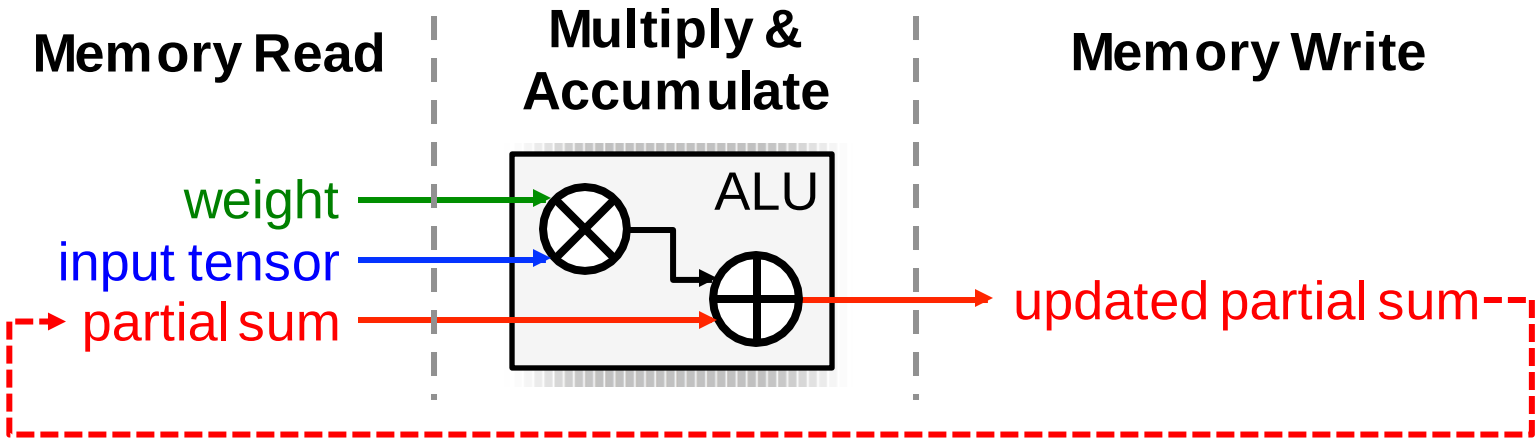
3. where to **store** data and how to **move** them around



The main source of headaches for DL Accelerator Architects!

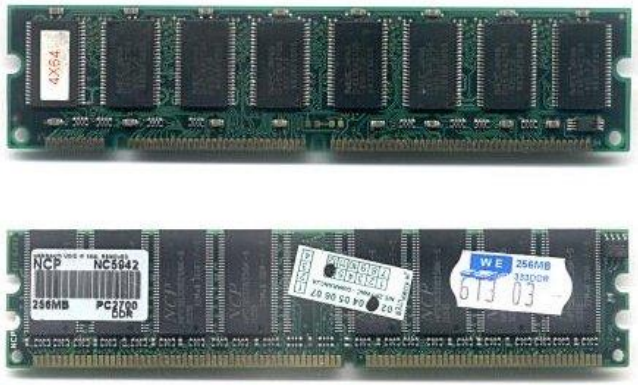
A Minimal Accelerator

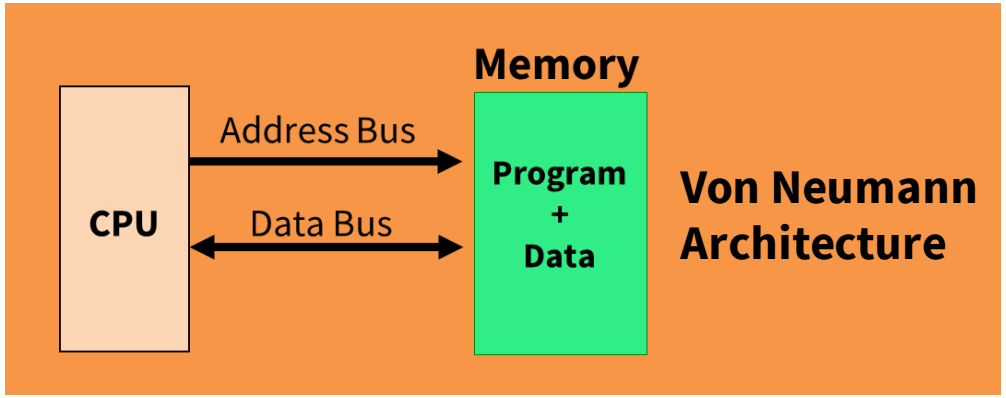
COMPUTE:



MEMORY:

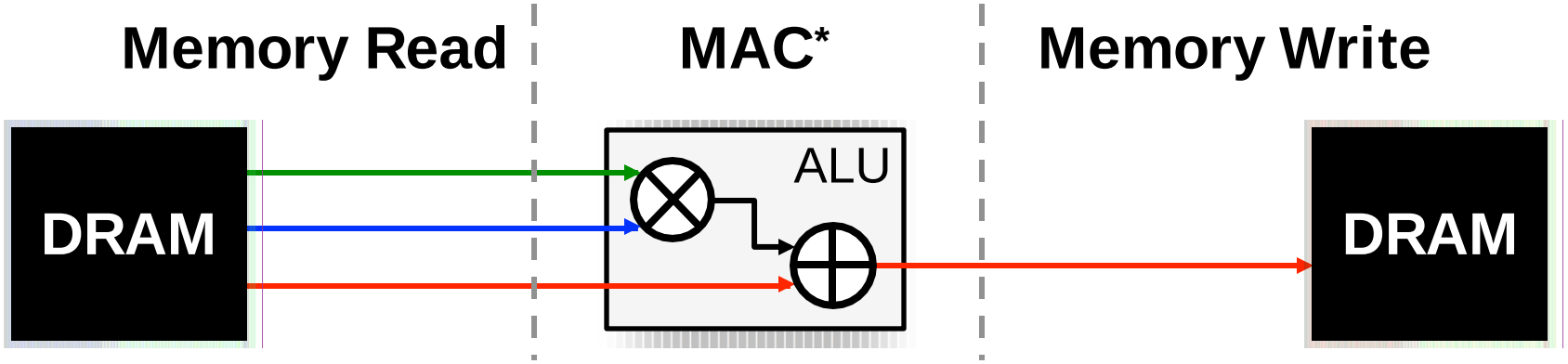
Off-Chip >10^{-9}J



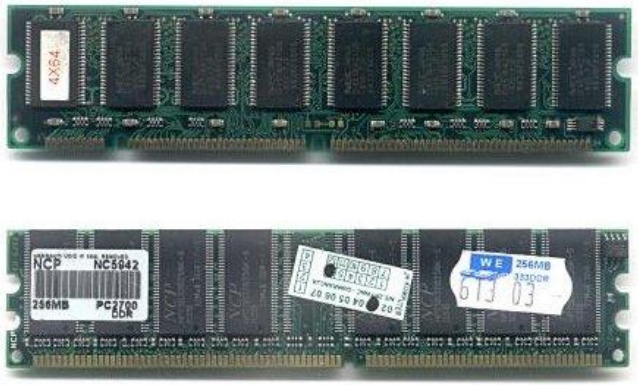


Non-Von Neumann… see the other talk!

Memory Access is the Bottleneck



Off-Chip >10^{-9}J



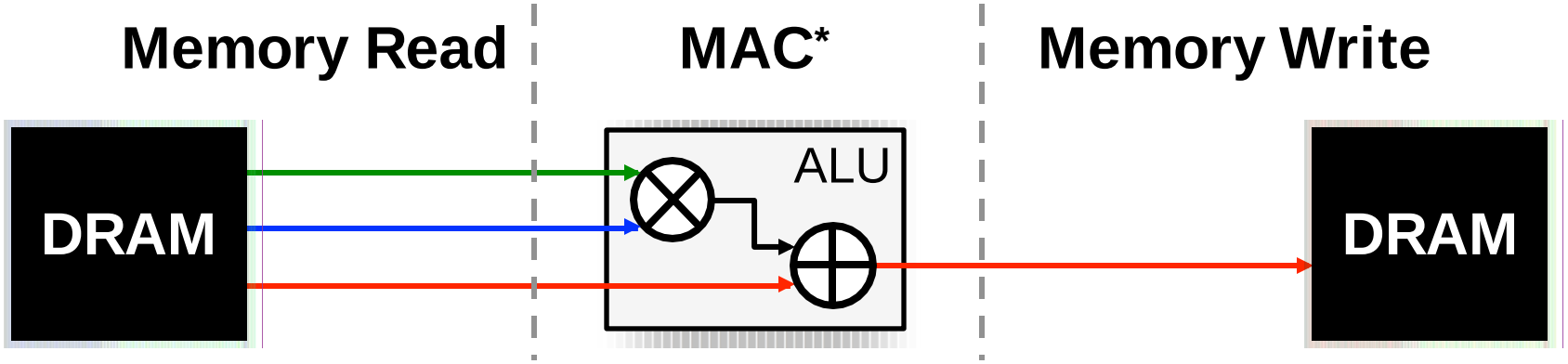
Worst Case: all memory R/W are **DRAM** accesses

AlexNet [NIPS 2012] has **724M** MACs

→ **2896M** DRAM accesses required

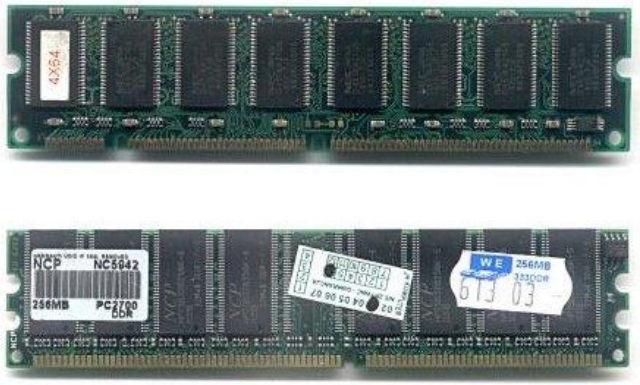
DRAM access 100-1000x less energy-efficient than on-chip access!

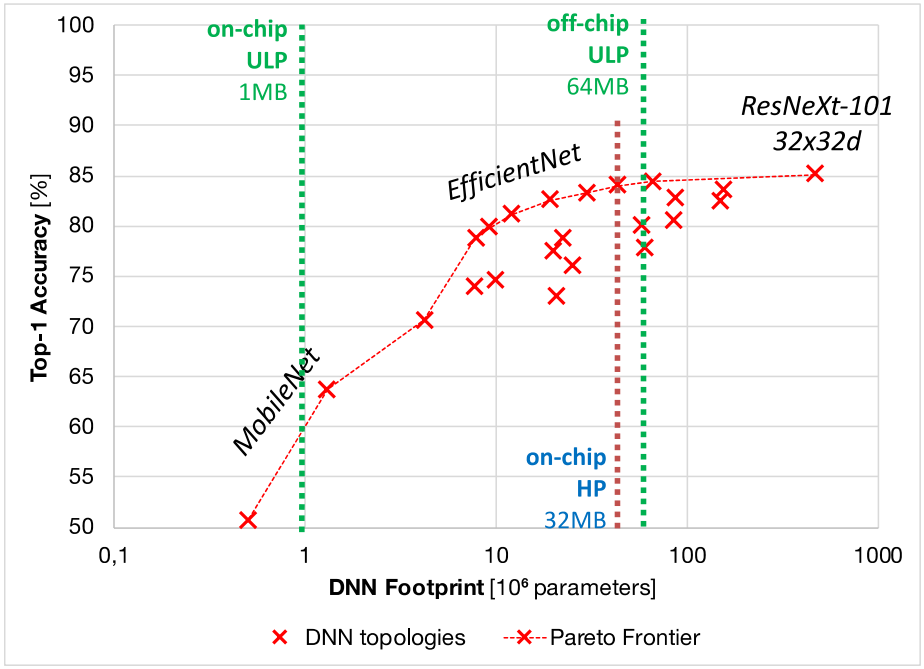
Memory Access is the Bottleneck



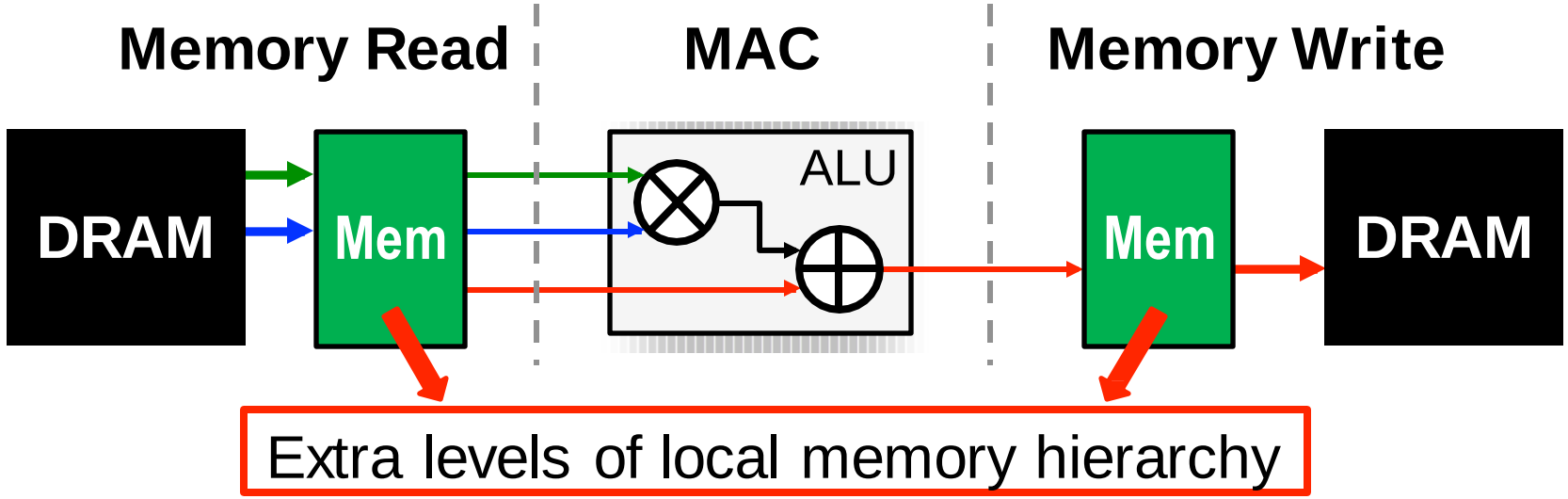
* multiply-and-accumulate

Off-Chip >10^{-9}J

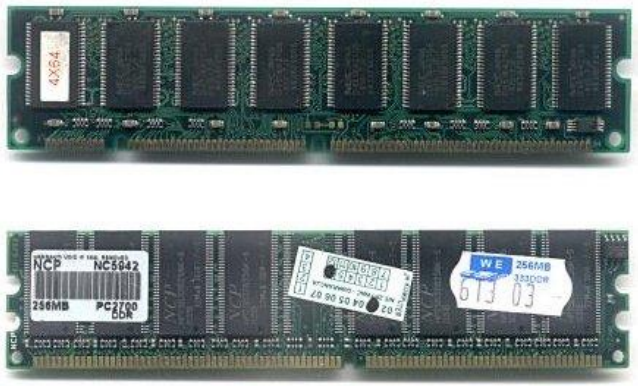




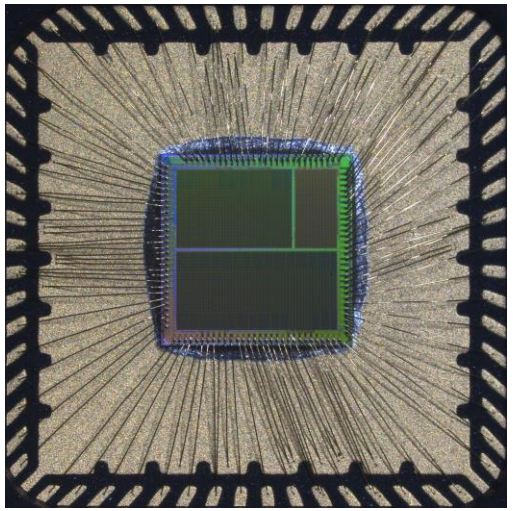
Memory Access is the Bottleneck



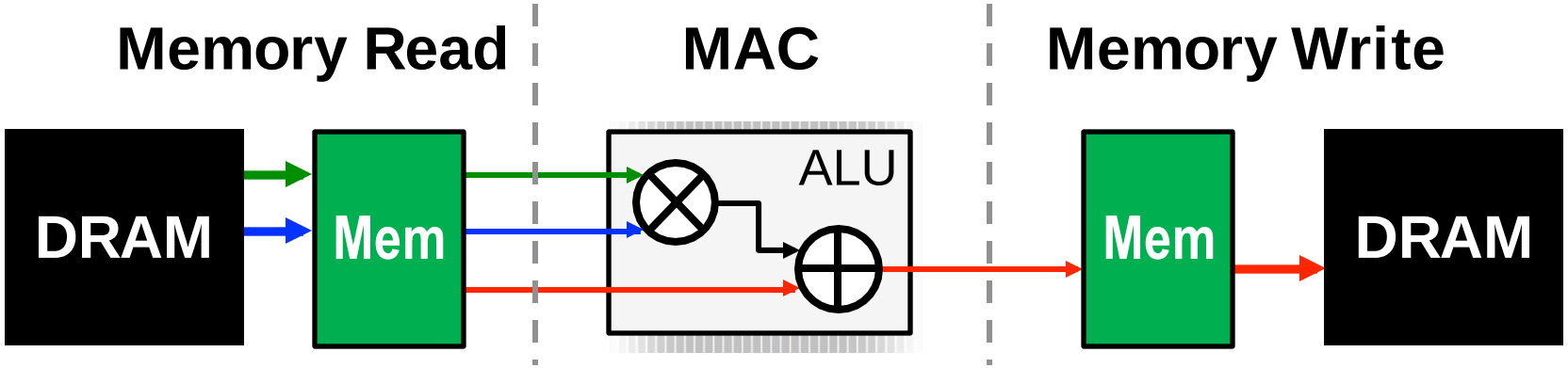
Off-Chip >10^{-9}J

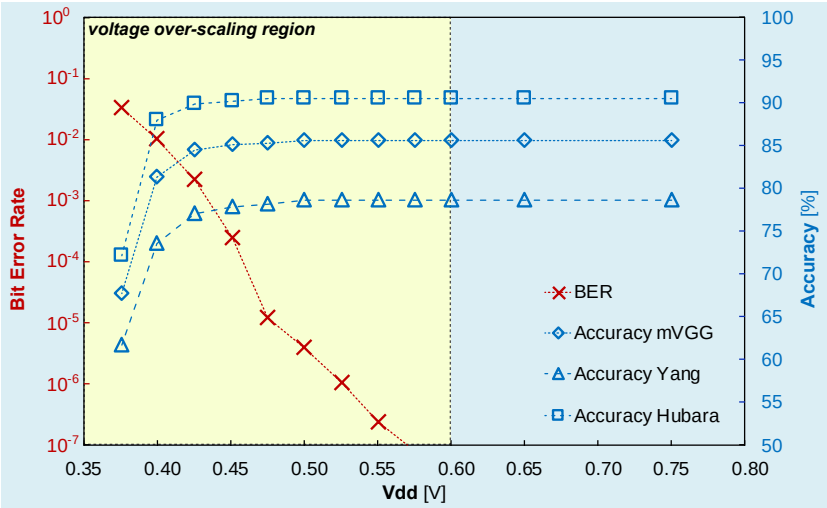


On-Chip ~10^{-11}J



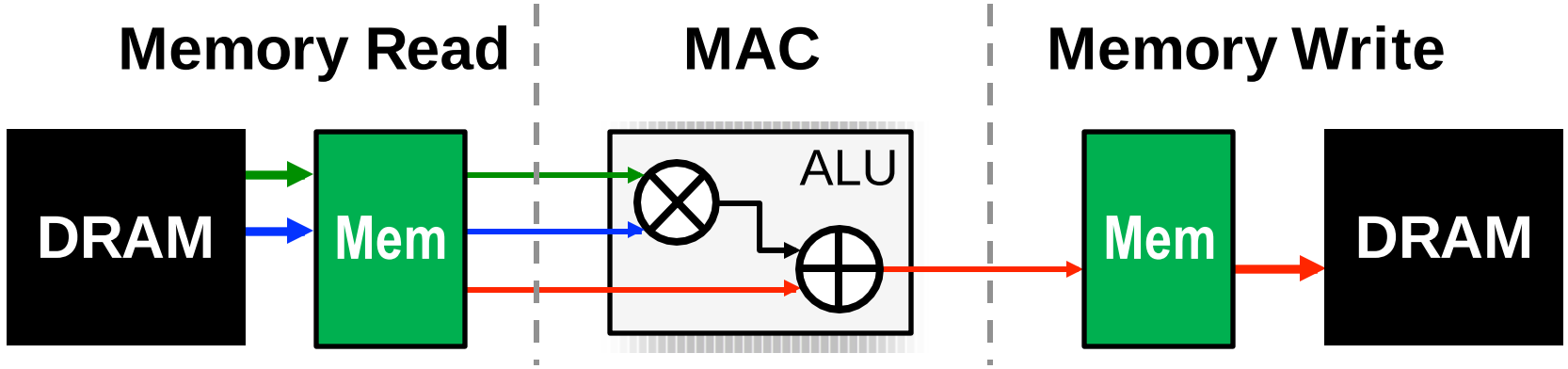
Memory Access is the Bottleneck

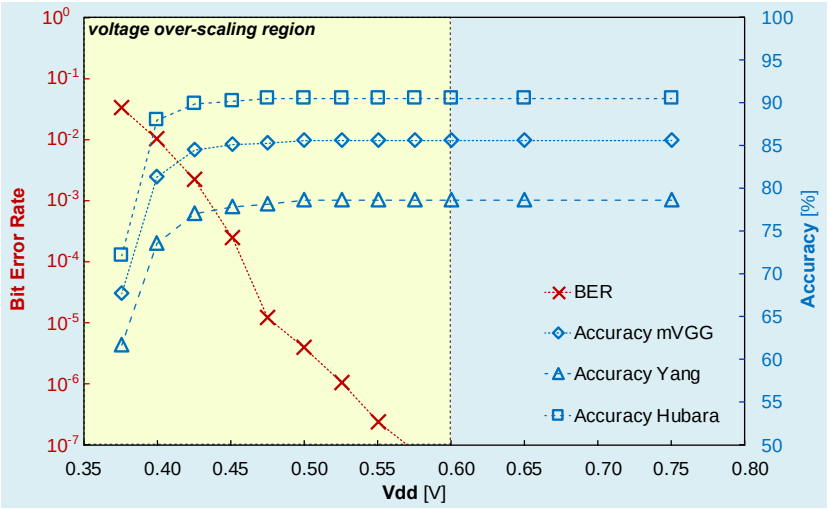




Reduce memory cost by **error resilience**

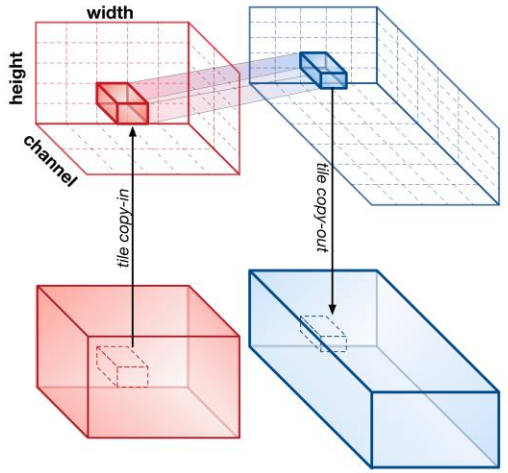
Memory Access is the Bottleneck





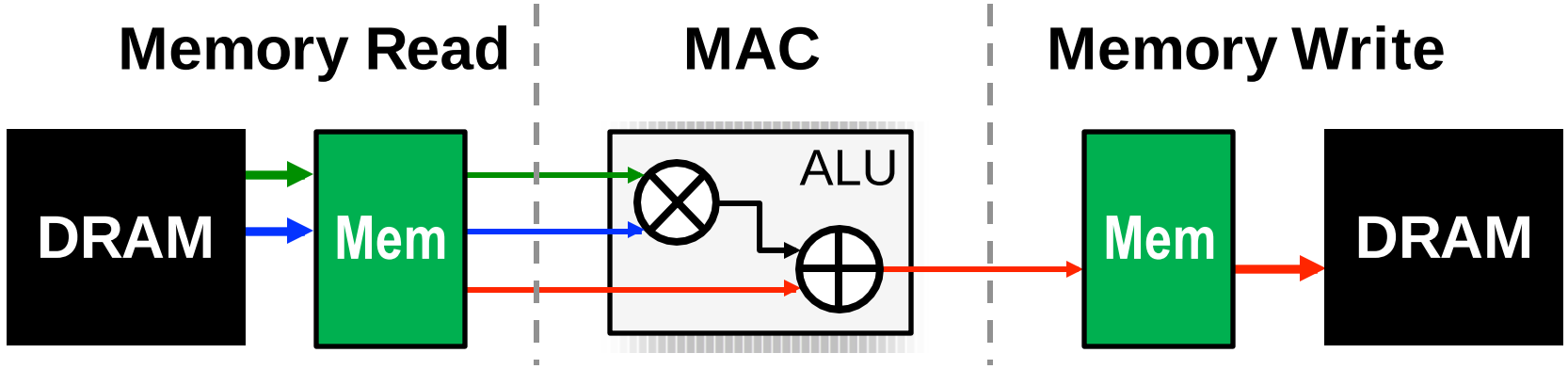
Reduce memory cost by **error resilience**

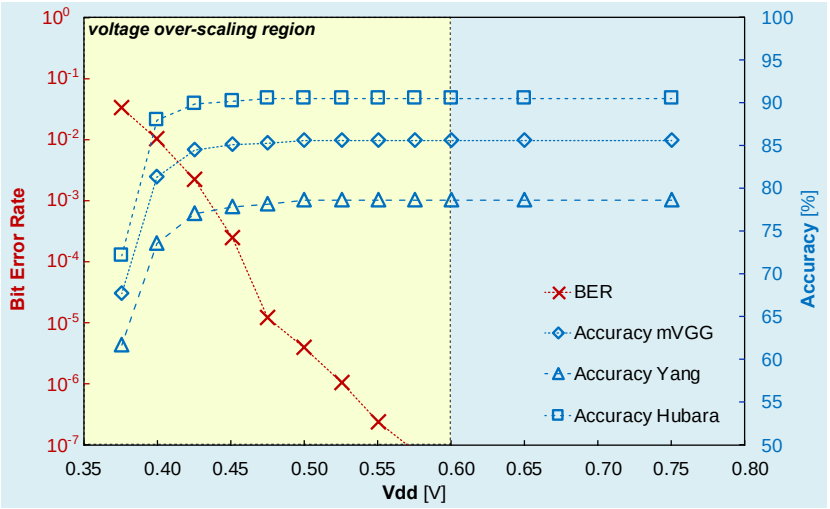
+



Reduce transfer cost by **data tiling**

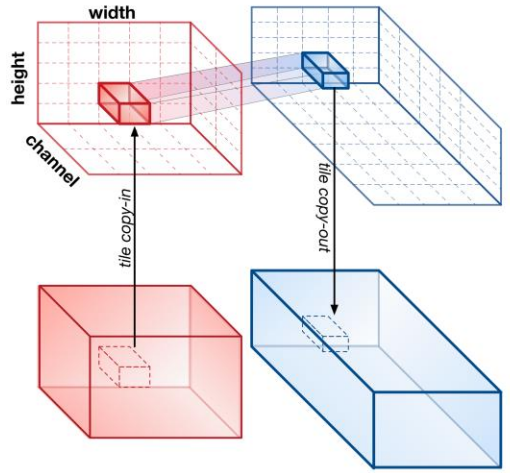
Memory Access is the Bottleneck





Reduce memory cost by **error resilience**

+



Reduce transfer cost by **data tiling**

+

Reduced precision

The Recipe for a Deep Learning Accelerator

1. Many **Multiply-Accumulate (MAC)** units to exploit parallelism

2. Flexible or Customized **on-chip memory organization** to keep as much data as possible on-chip, maximise its reuse…

3. Keep track of all **external memory** transfer overheads!

The Recipe for a Deep Learning Accelerator

1. Many **Multiply-Accumulate (MAC)** units to exploit parallelism

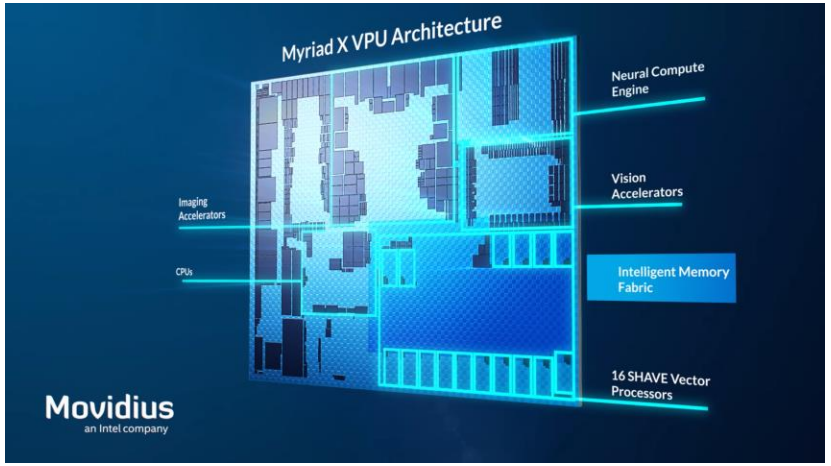
2. Flexible or Customized **on-chip memory organization** to keep as much data as possible on-chip, maximise its reuse…

3. Keep track of all **external memory** transfer overheads!

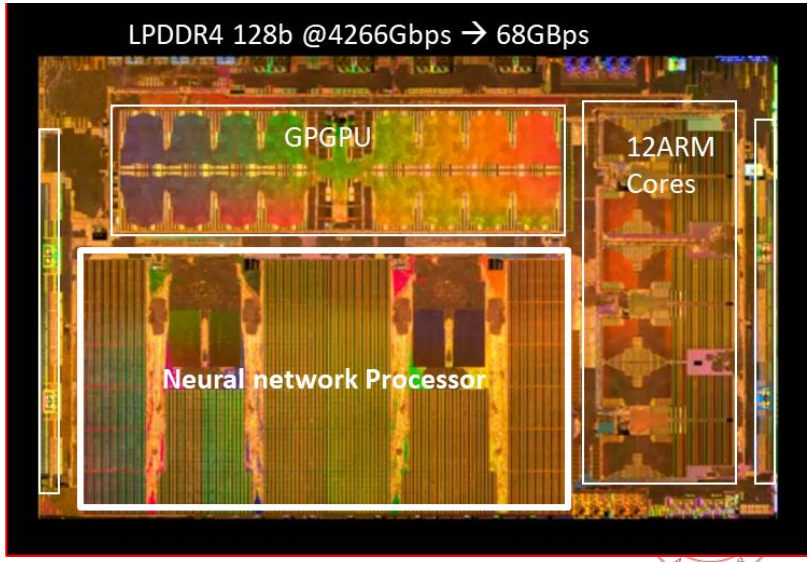
Custom architectures

Google Cloud TPU

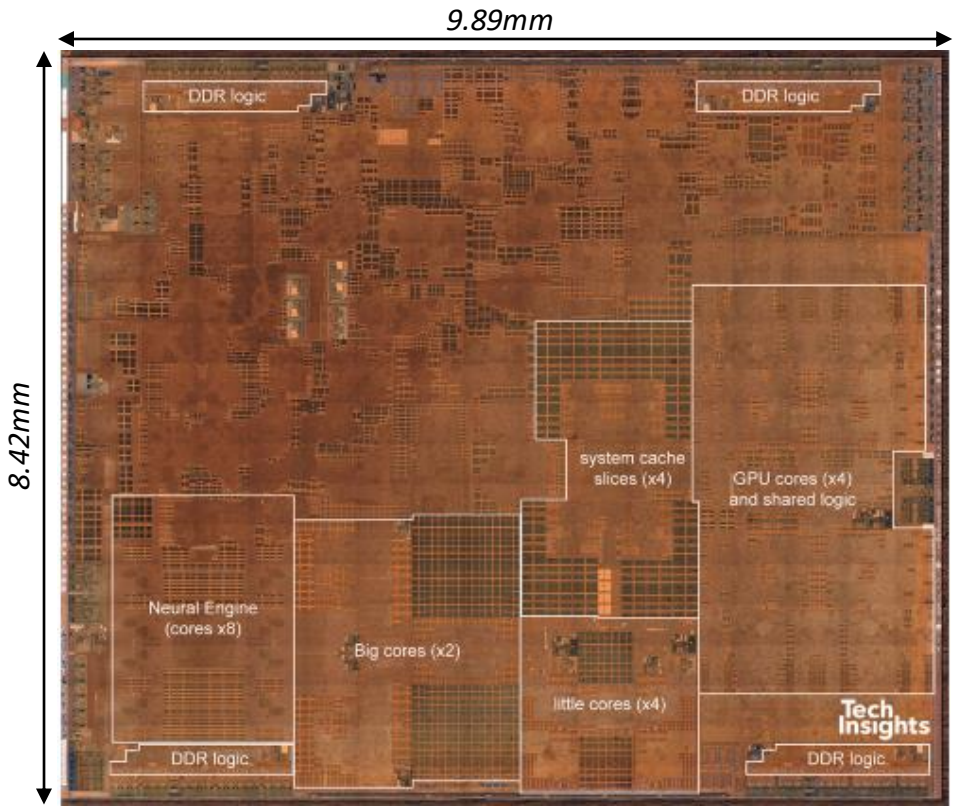
Intel/Movidius Myriad X



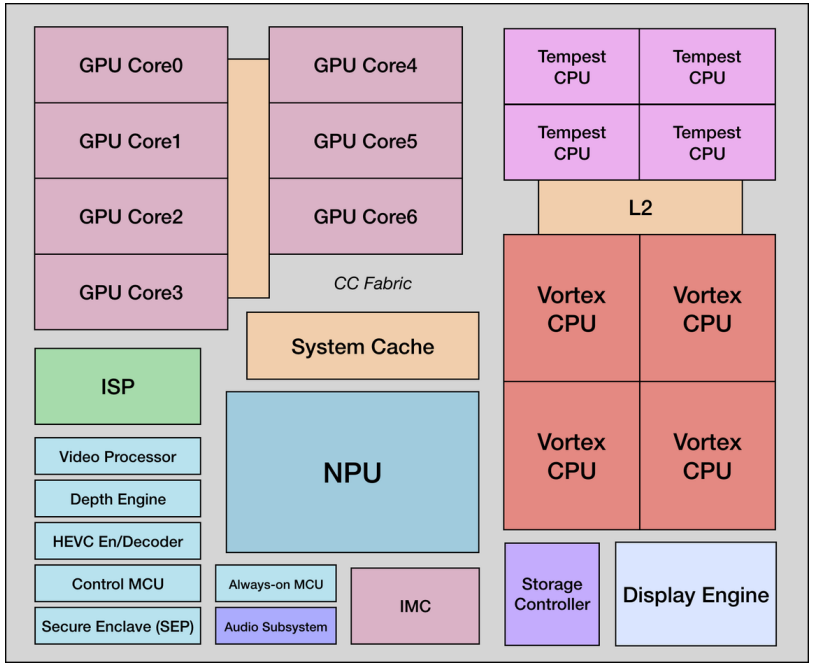
Tesla FSF Chip



The Recipe for a Deep Learning Accelerator



A12 (iPhone XS) – 7nm



A12X (iPad Pro 2018)

How many processors? A lot!

CPU: 2/4 "big" cores + 4 "small" cores

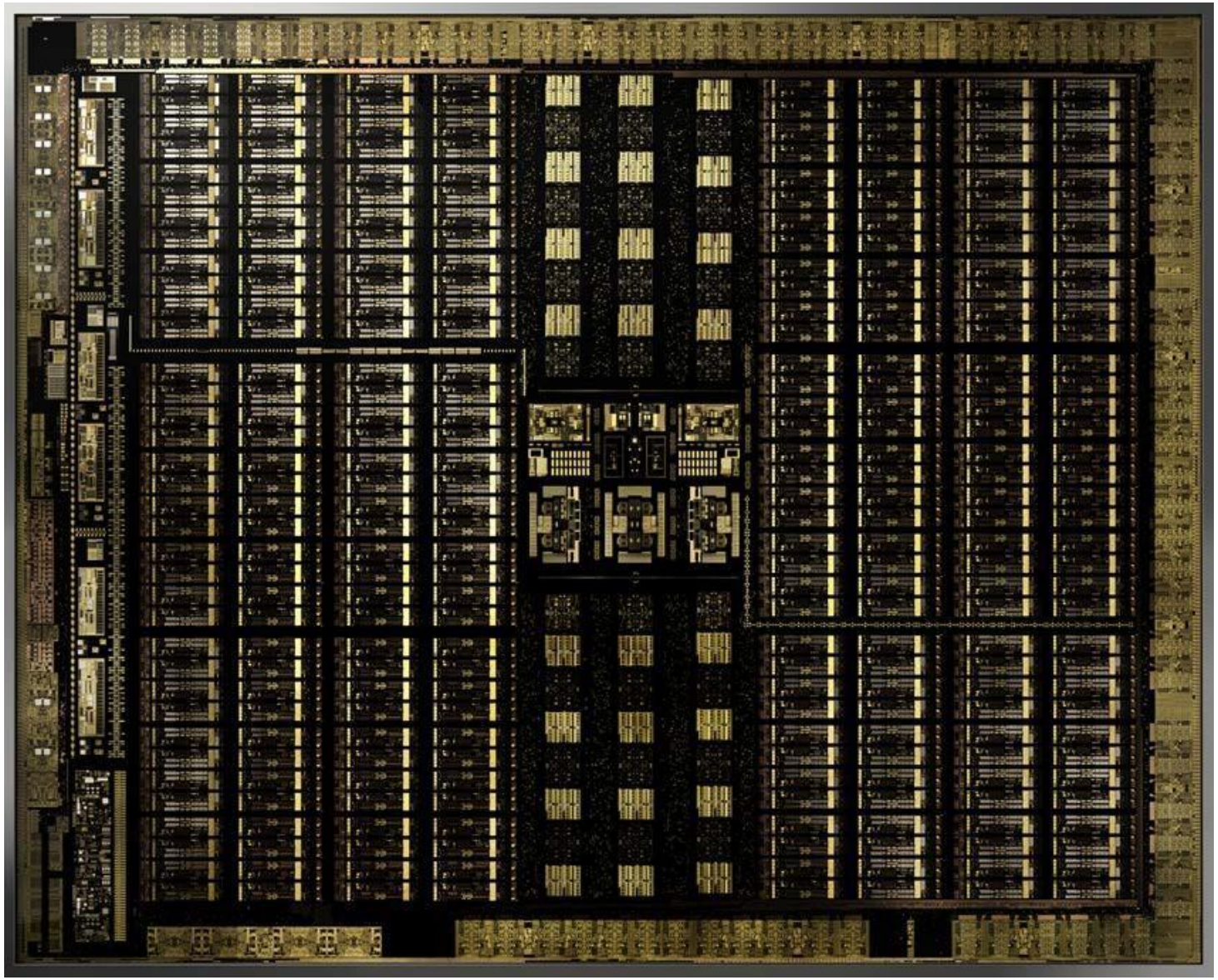
GPU: 4/6 cores

Accelerators: Neural Processing Unit (NPU) + Image Signal Processor (ISP)

MCUs: Control + Always-On

[WikiChip]

The Queen of Deep Learning Accelerators: the GPU



[NVIDIA Turing]

The Queen of Deep Learning Accelerators: the GPU

NVIDIA GV100

5120 cores
80 multiprocessors

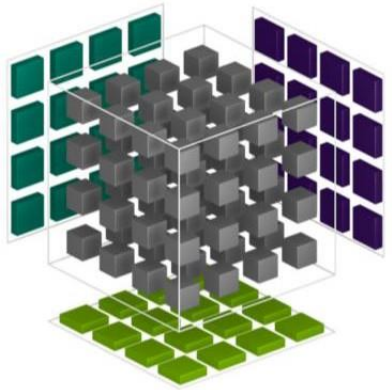
6MB L2 cache
per chip

128KB L1 cache
per multiprocessor

Tensor Cores

TENSOR CORE

Mixed Precision Matrix Math
4x4 matrices



$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 — FP16 — FP16 — FP16 or FP32

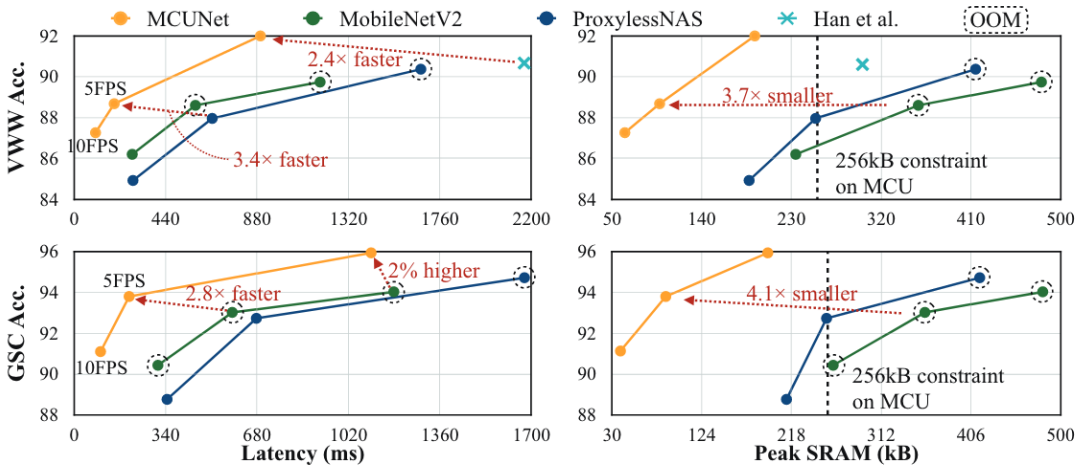
$$D = AB + C$$

There's Plenty of Room at the Bottom

- The relationship between **data representation**, **network topology**, and **perf/energy/memory** is not yet fully explored (particularly for tiny devices)!

- The **Von Neumann** model could be suboptimal: is it possible to sidestep memory in doing **Multiply-Adds**?

- Will future sophisticated AI algorithms show the same "good" properties of DNNS: **regularity**, **parallelism**, **resilience**?



(a) Trade-off: accuracy *v.s.* measured latency

(b) Trade-off: accuracy *v.s.* peak memory

[Lin et al., MCUNet: Tiny Deep Learning on IoT Devices]

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Prof. Francesco Conti

DEI – Università di Bologna

f.conti@unibo.it

www.unibo.it